Refuting Tarski and Gödel with a Sound Deductive Formalism

The conventional notion of a formal system is adapted to conform to the sound deductive inference model operating on finite strings. Finite strings stipulated to have the semantic value of Boolean true provide the sound deductive premises. Truth preserving finite string transformation rules provide the deductive inference. Sound deductive conclusions are the result of these finite string transformation rules.

The {domain of discourse} of the **Sound Deductive Formalism** (SDF) is the body of **Analytical_Knowledge** defined as follows: The set of knowledge that can be expressed using language and verified as true entirely on the basis of stipulated relations between expressions of language.

Validity and Soundness https://www.iep.utm.edu/val-snd/

A deductive argument is said to be *valid* if and only if it takes a form that makes it impossible for the premises to be true and the conclusion nevertheless to be false. Otherwise, a deductive argument is said to be *invalid*.

A deductive argument is *sound* if and only if it is both valid, and all of its premises are *actually true*. Otherwise, a deductive argument is *unsound*.

It seems self-evident that any formal system conforming to the above Sound Deductive Inference Model (SDIM) that applies truth preserving finite string transformation rules to a set of finite strings that are stipulated to have the semantic value of Boolean true would have a universal Truth(X) predicate on the basis of its universal Provable(X) predicate thus refuting both Tarski and Gödel for the domain of discourse of Analytical_Knowledge.

To provide a simple intuitive grasp of the **Sound Deductive Formalism** (SDF) we define a very simple formal system named **Simple_Arithmetic**.

All that **Simple_Arithmetic** does is evaluate relational_expressions comprised of a pair of arithmetic_expressions. These expressions have the exact same syntax as the “C” programming language. The arithmetic_expressions are limited to the operations of addition and multiplication of unsigned integer literals comprised entirely of the ASCII digits [0-9].

The only divergence from the “C” standard is that these unsigned integer literals are of arbitrary length and all arithmetic operations are performed directly on these strings of ASCII digits. This formal system would have a single Boolean Evaluate() function.

Evaluate(“((((2 + 3) * 7) + 9) == 44)”) evaluates to true which indicates that a set of finite string transformation rules derives: “44” from: “(((2 + 3) * 7) + 9)” thus satisfying: “==”.

In other words the finite string transformation rules that evaluate that the above expression to true are the formal proof that the above expession is true.

Axioms, rules-of-inference, syntax, and truth conditional semantics are all fully integrated together into the single operation of finite string transformation rules.

When a finite string X evaluates to True(X) we know that it has been proven true on the basis of finite string transformation rules. This also ensures that X was a WFF.

When a finite string X evaluates to ~True(X) we know that it is not provable the basis of finite string transformation rules. This might be because X was not a WFF.